Créer un compte Se connecter

Article Discussion Lire Modifier le code Historique Rechercher

WIKIPÉDIA
L'encyclopédie libre

Navigation
Accueil
Portails thématiques
Article au hasard
Contact

Contribuer
Débuter sur Wikipédia
Aide
Communauté
Modifications récentes
Faire un don

Imprimer / exporter
Créer un livre
Télécharger comme PDF
Version imprimable

Outils
Pages liées
Suivi des pages liées
Importer un fichier
Pages spéciales
Adresse de cette version
Information sur la page
Élément sur Wikidata
Citer cette page

Autres langues
Čeština
Deutsch
English
Modifier les liens

Permissions UNIX

Les **permissions UNIX** constituent un système simple de définition des droits d'accès aux ressources, représentées par des fichiers disponibles sur un système informatique. Elles restent le moyen le plus utilisé pour définir les droits des utilisateurs sur les systèmes de type UNIX.

Sommaire

1 Introduction
1.1 Notion d'utilisateur (*user*)
1.2 Groupe
1.3 Propriété
1.4 Droits d'accès à un fichier
1.5 Norme POSIX
2 Fonctionnement
2.1 Les différents droits
2.2 Représentation des droits
2.3 Utilisation
3 Droits étendus
3.1 Droit SUID
3.1.1 Définition
3.1.2 Notation
3.1.3 Valeur
3.1.4 Exemple d'utilisation
3.1.5 Protection contre une attaque de type *sushi*
3.2 Droit SGID
3.2.1 Définition
3.2.1.1 Fichiers
3.2.1.2 Répertoires
3.2.2 Notation
3.2.3 Valeur
3.3 *Sticky Bit*
3.3.1 Définition
3.3.2 Notation
3.3.3 Valeur
3.3.4 Exemple d'utilisation
4 Ajout de la gestion des listes de contrôle d'accès
5 Comparaison avec le NTFS de Microsoft
6 Notes
7 Références

Introduction [modifier le code]

Notion d'utilisateur (*user*) [modifier le code]

Toute entité (personne physique ou programme particulier) devant interagir avec un système UNIX est authentifié sur cet ordinateur par un utilisateur ou *user*. Ceci permet d'identifier un acteur sur un système UNIX. Un utilisateur est reconnu par un nom unique et un numéro unique (la correspondance nom/numéro est stockée dans le fichier */etc/passwd*).

Tous les utilisateurs UNIX n'ont pas les mêmes droits d'accès à l'ordinateur (ils ne peuvent pas tous faire la même chose), et ceci simplement pour des raisons de sécurité et d'administration. Par exemple, pour éviter tout problème sur Internet, l'utilisateur qui gère le serveur HTTP n'a pas le droit d'exécuter des commandes localement, pour éviter que le serveur ne puisse le faire.

Certains utilisateurs ne peuvent en effet pas s'authentifier sur l'ordinateur et accéder à un interpréteur de commandes. Cela ne veut toutefois pas dire qu'ils ne peuvent rien faire sur l'ordinateur : il leur est possible de lire ou écrire des fichiers mais cela nécessite que le super-utilisateur (voir plus bas) démarre un programme pour cet utilisateur. Ce mécanisme est généralement utilisé pour les démons : le super utilisateur démarre le démon et pour éviter que ce dernier ne puisse faire tout et n'importe quoi sur la machine, il est par exemple attribué à l'utilisateur bin.

Sur tout système UNIX, il y a un super-utilisateur, généralement appelé *root*, qui a tous les pouvoirs. Il peut accéder librement à toutes les ressources de l'ordinateur, y compris à la place d'un autre utilisateur, c'est-à-dire sous son

identité. En général, du moins sur les systèmes de production, seul l'administrateur système possède le mot de passe *root*. L'utilisateur *root* porte le numéro 0.

Groupe [modifier le code]

Un utilisateur UNIX appartient à un ou plusieurs groupes. Les groupes servent à rassembler des utilisateurs afin de leur attribuer des droits communs. Par exemple, sur un système doté d'une carte son, il y a souvent un groupe *audio* qui regroupe les utilisateurs autorisés à en faire usage.

Propriété [modifier le code]

Tout fichier UNIX possède un propriétaire. Au départ, c'est l'utilisateur qui a créé le fichier mais "root" peut le « donner » à un autre utilisateur. Seul le propriétaire du fichier et le super utilisateur (*root*) peuvent changer les droits.

Traditionnellement, les Unix "System V" permettent au propriétaire de changer le possesseur d'un fichier, mais dans la tradition BSD et sous Linux, seul "root" peut changer le propriétaire d'un fichier[1], notamment pour éviter qu'un utilisateur n'échappe aux quotas disque en donnant ses fichiers à un autre utilisateur.

Un fichier UNIX appartient aussi à un groupe. Ceci donne pleinement son sens à la notion de groupe. On définit ainsi les actions du groupe sur ce fichier. Ce groupe est souvent le groupe d'appartenance du propriétaire, mais ce n'est pas obligatoire. Tout dépend en fait de ce qu'on veut faire. On peut imaginer un scénario de délégation d'administration : le super utilisateur est propriétaire d'un fichier de configuration, mais autorise tous les utilisateurs du groupe admin (les administrateurs) à modifier ce fichier. Le fichier en question aura donc root comme propriétaire et appartiendra au groupe admin.

Rappelons que les répertoires sous UNIX sont aussi des fichiers. Les droits sur les répertoires (mais aussi les périphériques, etc.) fonctionnent exactement de la même façon que sur des fichiers ordinaires.

Droits d'accès à un fichier [modifier le code]

À chaque fichier est associée une liste de permissions, qui déterminent ce que chaque utilisateur a le droit de faire du fichier.

Norme POSIX [modifier le code]

Les permissions d'accès aux fichiers dans la norme POSIX sont inspirées des permissions d'accès UNIX.

Fonctionnement [modifier le code]

Les différents droits [modifier le code]

Les droits sur un fichier UNIX s'attribuent sur trois « actions » différentes possibles :

- la lecture (r) : on peut par exemple lire le fichier avec un logiciel. Lorsque ce droit est alloué à un répertoire, il autorise l'affichage du contenu du répertoire (la liste des fichiers présents à la racine de ce répertoire).
- l'écriture (w) : on peut modifier le fichier et le vider de son contenu. Lorsque ce droit est alloué à un répertoire, il autorise la création, la suppression et le changement de nom des fichiers qu'il contient, quels que soient les droits d'accès des fichiers de ce répertoire (même s'ils ne possèdent pas eux-mêmes le droit en écriture). Néanmoins le droit spécial *sticky bit* permet de passer outre ce comportement.
- l'exécution (x) : on peut exécuter le fichier s'il est prévu pour, c'est-à-dire si c'est un fichier exécutable. Lorsque ce droit est attribué à un répertoire, il autorise l'accès (ou ouverture) au répertoire.

On appelle parfois r, w et x des « flags » ou « drapeaux ». Sur un fichier donné, ces 3 flags doivent être définis pour son propriétaire, son groupe, mais aussi les autres utilisateurs (différents du propriétaire et n'appartenant pas au groupe).

Seuls *root* et le propriétaire d'un fichier peuvent changer ses permissions d'accès.

Représentation des droits [modifier le code]

Cet ensemble de 3 droits sur 3 entités se représente généralement de la façon suivante : on écrit côte à côte les droits r, w puis x respectivement pour le propriétaire (u), le groupe (g) et les autres utilisateurs (o). Les codes u, g et o (u comme *user*, g comme *group* et o comme *others*) sont utilisés par les commandes UNIX qui permettent d'attribuer les droits et l'appartenance des fichiers. Lorsqu'un flag est attribué à une entité, on écrit ce flag (r, w ou x), et lorsqu'il n'est pas attribué, on écrit un '-'. Par exemple,

```
rwxr-xr--
\ /\ /\ /
 v  v  v
 |  |  droits des autres utilisateurs (o)
```

```
  |  |
  |  droits des utilisateurs appartenant au groupe (g)
  |
droits du propriétaire (u)
```

signifie que le propriétaire peut lire, écrire et exécuter le fichier, mais que les utilisateurs du groupe attribué au fichier ne peuvent que le lire et l'exécuter, et enfin que les autres utilisateurs ne peuvent que lire le fichier.

Une autre manière de représenter ces droits est sous forme binaire grâce à une clef numérique fondée sur la correspondance entre un nombre décimal et son expression binaire :

- 0 = 000
- 1 = 001
- 2 = 010
- 3 = 011
- 4 = 100
- 5 = 101
- 6 = 110
- 7 = 111

À l'expression binaire en trois caractères sont associés les 3 types de droits (r w x) ; il suffit donc de déclarer pour chacune des catégories d'utilisateur (user, group, others) un chiffre entre 0 et 7 auquel correspond une séquence de droits d'accès. Par exemple :

- 777 donne 111 111 111 soit r w x r w x r w x
- 605 donne 110 000 101 soit r w - - - - r - x
- 644 donne 110 100 100 soit r w - r - - r - -
- 666 donne 110 110 110 soit r w - r w - r w -

Une astuce permet d'associer rapidement une valeur décimale à la séquence de droits souhaitée. Il suffit d'attribuer les valeurs suivantes pour chaque type de droit :

- lecture (r) correspond à 4
- écriture (w) correspond à 2
- exécution (x) correspond à 1

Puis on additionne ces valeurs selon qu'on veuille ou non attribuer le droit en correspondant.

Ainsi, rwx « vaut » 7 (4+2+1), r-x « vaut » 5 (4+1) et r-- « vaut » 4. Les droits complets (rwxr-xr--) sont donc équivalent à 754. Une manière directe d'attribuer les droits est de les écrire sous cette forme et d'utiliser le code à 3 chiffres résultant avec chmod (voir ci-après).

Utilisation [modifier le code]

Pour voir quels droits sont attribués à un fichier, il suffit de taper la commande ls -l nom_du_fichier :

```
# ls -l toto
-rwxr-xr--    1 user     group     12345 Nov 15 09:19 toto
```

La sortie signifie que le fichier toto (de taille 12345) appartient à « user », qu'on lui a attribué le groupe « group », et que les droits sont rwxr-xr--. On remarque qu'il y a en fait 10 caractères sur la zone de droits. Le premier - n'est pas un droit, c'est un caractère réservé pour indiquer le type de fichier. Il peut prendre les valeurs suivantes :

- `d` : répertoire
- `l` : lien symbolique
- `c` : périphérique de type caractère
- `b` : périphérique de type bloc
- `p` : fifo
- `s` : socket
- `-` : fichier classique

Le changement de droits s'effectue avec la commande chmod ; le changement de propriétaire ou de groupe, à l'aide de la commande chown.

Changer les droits peut s'effectuer également simplement à partir du nombre à 3 chiffres calculé comme précédemment. Ainsi, pour attribuer les droits r-xr-xr-x (i.e. 555), il suffit d'exécuter :

```
chmod 555 nom_du_fichier
```

Droits étendus [modifier le code]

Il existe d'autres droits spéciaux, rendant possible une gestion plus poussée des permissions.

Droit SUID [modifier le code]

Article détaillé : setuid.

Définition [modifier le code]

Ce droit s'applique aux fichiers exécutables, il permet d'allouer temporairement à un utilisateur les droits du propriétaire du fichier, durant son exécution. En effet, lorsqu'un programme est exécuté par un utilisateur, les tâches qu'il accomplira seront restreintes par ses propres droits, qui s'appliquent donc au programme. Lorsque le droit SUID est appliqué à un exécutable et qu'un utilisateur quelconque l'exécute, le programme détiendra alors les droits du propriétaire du fichier durant son exécution. Bien sûr, un utilisateur ne peut jouir du droit SUID que s'il détient par ailleurs les droits d'exécution du programme. Ce droit est utilisé lorsqu'une tâche, bien que légitime pour un utilisateur classique, nécessite des droits supplémentaires (généralement ceux de *root*). Il est donc à utiliser avec précaution. Pour des partitions supplémentaires, il faut activer le bit suid pour pouvoir l'utiliser en le spécifiant dans les options des partitions concernés dans le fichier fstab.

Notation [modifier le code]

Son flag est la lettre s ou S qui vient remplacer le x du propriétaire. La majuscule sur le flag permet de voir le flag x (droit d'exécution du propriétaire) caché par le droit SUID. C'est un s si le droit d'exécution du propriétaire est présent, ou un S sinon. Il se place donc comme ceci :

```
---s------  ou  ---S------
```

Un fichier avec les droits

```
-rwxr-xr-x
```

auquel on ajoute le droit SUID aura donc la notation

```
-rwsr-xr-x
```

Valeur [modifier le code]

Le droit SUID possède la valeur octale 4000.
Exemple : - r w s r - x r - x correspond à 4755.

Exemple d'utilisation [modifier le code]

Le droit SUID est utilisé pour le programme *mount* (généralement localisé dans /bin) :

```
-rwsr-xr-x  1 root bin 68804 /bin/mount
```

Protection contre une attaque de type *sushi* [modifier le code]

Sur linux, des restrictions de droit ont été faites sur le drapeau SUID :

- pour un fichier de script, l'exécution se fera sans tenir compte du drapeau SUID
- Il semble de plus que si un utilisateur autre que le propriétaire modifie le fichier, alors le bit SUID est remis à 0.

Il n'est pas sûr que les versions d'Unix actuelles aient implémenté cette sécurité ; dans ce cas, cela ouvre la porte à une attaque qui est appelée *sushi* (*su shell*) : l'utilisateur *root* crée un fichier de shell avec le bit SUID en laissant les droits en écriture à d'autres utilisateurs ; l'un d'eux pourra modifier le fichier et exécuter tout ce qu'il voudra avec les droits de root.

Droit SGID [modifier le code]

Définition [modifier le code]

Fichiers [modifier le code]

Ce droit fonctionne comme le droit SUID, mais appliqué aux groupes. Il donne à un utilisateur les droits du groupe auquel appartient le propriétaire de l'exécutable et non plus les droits du propriétaire.

Répertoires [modifier le code]

Ce droit a une tout autre utilisation s'il est appliqué à un répertoire. Normalement, lorsqu'un fichier est créé par un utilisateur, il en est propriétaire, et un groupe par défaut lui est appliqué (généralement *users* si le fichier a été créé par un utilisateur, et *root* ou *wheel* s'il a été créé par *root*). Cependant, lorsqu'un fichier est créé dans un répertoire portant le droit SGID, alors ce fichier se verra attribuer par défaut le groupe du répertoire. De plus, si c'est un autre répertoire qui est créé dans le répertoire portant le droit SGID, ce sous-répertoire portera également ce droit.

Notation [modifier le code]

Comme le droit SUID, il vient remplacer le flag x, mais celui des droits du groupe, et la majuscule indique le droit d'exécution x caché, elle est présente s'il n'est pas alloué.

```
------s---  ou  ------S---
```

Un fichier avec les droits

```
-rwxr--r--
```

auquel on ajoute le droit SGID aura donc la notation

```
-rwxr-Sr--
```

Valeur [modifier le code]

Le droit SGID possède la valeur octale 2000.
Exemple : - r w x r - S r - - correspond à 2744.

Sticky Bit [modifier le code]

Ce droit (traduction *bit collant*) est utilisé pour manier de façon plus subtile les droits d'écriture d'un répertoire. En effet, le droit d'écriture signifie que l'on peut créer et supprimer les fichiers de ce répertoire. Le sticky bit permet de faire la différence entre les deux droits.

Définition [modifier le code]

Lorsque ce droit est positionné sur un répertoire, il interdit la suppression d'un fichier qu'il contient à tout utilisateur autre que le propriétaire du fichier. Néanmoins, il est toujours possible pour un utilisateur possédant les droits d'écriture sur ce fichier de le modifier (par exemple de le transformer en un fichier vide). La création de nouveaux fichiers est toujours possible pour tous les utilisateurs possédant le droit d'écriture sur ce répertoire. Remarquons que c'est le même principe que les pages de Wikipédia, elles sont modifiables mais on ne peut pas les supprimer.

Pour les fichiers, l'utilisation est tout autre. Le bit collant (*sticky bit*) indique alors que ce fichier doit encore rester en mémoire vive après son exécution. Le but était, à l'époque, d'améliorer les performances en évitant de charger/décharger un fichier de la mémoire (par exemple, un exécutable ou une bibliothèque logicielle). Le terme *sticky* (collant) voulait dire que le fichier restait collé en mémoire.

Notation [modifier le code]

Son flag est le t ou T, qui vient remplacer le droit d'exécution x des autres utilisateurs que le propriétaire et ceux appartenant au groupe du fichier, de la même façon que les droits SUID et SGID. La majuscule fonctionne aussi de la même façon, elle est présente si le droit d'exécution x caché n'est pas présent.

```
---------t ou ---------T
```

Un fichier avec les droits

```
-rwxr-xr-x
```

auquel on ajoute le droit sticky bit aura donc la notation

```
-rwxr-xr-t
```

Valeur [modifier le code]

Le droit sticky bit possède la valeur octale 1000.
Exemple : - r w x r - x r - t correspond à 1755.

Exemple d'utilisation [modifier le code]

Sur certains systèmes UNIX, le sticky bit est utilisé pour le répertoire /tmp, qui doit être accessible en écriture par tous les utilisateurs, sans que ceux-ci se suppriment leurs fichiers les uns les autres :

```
drwxrwxrwt   86 root root  20480 tmp
```

Ajout de la gestion des listes de contrôle d'accès [modifier le code]

Avec les permissions UNIX, il n'est pas possible d'avoir une gestion des listes de contrôle d'accès très fine : on ne peut seulement donner des droits particuliers qu'à :

- 1 utilisateur ;
- 1 groupe d'utilisateurs ;
- ceux qui ne sont ni l'un, ni l'autre.

Il est possible d'y remédier en rajoutant un logiciel supplémentaire, pour la gestion des listes de contrôle d'accès. Dans le cas particulier d'une des distributions GNU/Linux, cela s'obtient en installant le paquet correspondant (exemple : paquet *acl* dans les archives debian).

Un fichier disposant d'une liste d'ACLs sera présenté avec un "+" en bout de ligne

```
-rwxr--r--+
```

Comparaison avec le NTFS de Microsoft [modifier le code]

Les permissions NTFS de Microsoft et les permissions d'accès UNIX sont conformes à la norme POSIX, mais sont profondément différentes :

1. NTFS inclut la gestion des listes de Contrôle d'accès (ACL) : les systèmes de type UNIX ne disposent pas tous des ACL. Les ACL permettent de contrôler plus finement les droits de bases, en ayant plus de découpages au niveau des droits (lecture, écriture, etc.), et de définir les droits pour chaque utilisateur et chaque groupe, si on le souhaite ;
2. Sur NTFS, il y a 2 permissions qui n'existent pas explicitement sur les systèmes de type UNIX :
   - le droit de s'approprier un fichier ;
   - le droit de changer les droits ;
3. Le droit de modification NTFS ne permet pas la suppression du fichier comme sous UNIX, il y a donc un droit spécifique de suppression du fichier.

La complexité de la gestion des droits sous Windows n'est pas forcément un avantage : en augmentant la complexité, on peut diminuer la maîtrise et la compréhension des interactions entre les droits du système par un humain. Une gestion simple de la sécurité peut donner un meilleur sentiment de maîtrise du système mais empêche une certaine finesse.

Notes [modifier le code]

1. ↑ Why can only root chown files?

Références [modifier le code]

- Les manuels UNIX (section 1) : man chmod et man chown
- chown --help
- chmod --help
- Droit et permission d'accès aux fichiers sous Linux
- Page de manuel en français de la commande CHMOD
- Page de manuel en français de la commande CHOWN

v · d · m	Contrôle d'accès en sécurité informatique
Types de contrôle d'accès	Obligatoire · Discrétionnaire · À base de rôles · À base de règles · À base de treillis · À base d'organisations
Modèles de contrôle d'accès	Bell-La Padula · Biba · Brewer et Nash · Clark-Wilson · Graham-Denning · Protection Take-grant

Portail de la sécurité informatique Portail GNU/Linux

Catégories : Unix | Contrôle d'accès informatique

Dernière modification de cette page le 9 avril 2014 à 23:52.

Droit d'auteur : les textes sont disponibles sous licence Creative Commons paternité partage à l'identique ; d'autres conditions peuvent s'appliquer. Voyez les conditions d'utilisation pour plus de détails, ainsi que les crédits graphiques. En cas de réutilisation des textes de cette page, voyez comment citer les auteurs et mentionner la licence.
Wikipedia® est une marque déposée de la Wikimedia Foundation, Inc., organisation de bienfaisance régie par le paragraphe 501(c)(3) du code fiscal des États-Unis.

Politique de confidentialité À propos de Wikipédia Avertissements Développeurs Version mobile